EXHIBIT 2

-------- Original Message --------
Subject: "specification" for cache regeneration
Date: Fri, 07 Jan 2000 20:01:46 -0600
From: Conleth O'Connell <cso@vignette.com>
To: mkm@vignette.com, mark@vignette.com, markm@vignette.com
CC: dls@vignette.com

Mike, Mark M.,

I'm attaching the "spec" Mark S. and I pulled together. Mark M/ has already read through it once, so I'm hoping it is clear enough. If not I'm available to talk about it. If any other issues come up as you go through the implementation (short-term vs. what is needed in the product), feel free to modify the document (use change bars), so we can review those things going forward.

What's important in this release is enough to support them without worrying about bcaps support (or language) on solaris. I'm trying to find out from robin if they are using any PZN functionality to help simplify the ctld protocol being duplicated.

If you have any questions, email me (here and home: cso@austin.rr.com), call me at home: 335-5080 or page me (873-6420)

Con

Cache management regeneration design

Mark and Con, et al

1 Problem characterization

1.1 Ctld overload

1.2 Web server resource usage

1.3 Disk thrashing (web site weirdness b/c backup files used)

1.4 database hit b/c of the new requests

1.5 Underlying cause: no file in docroot to serve

2 Requirements

2.1 Must support clear_cache from templates

2.2 Must support flushcache program task requests

2.3 Must support template modifications

2.4 Must deal with template variations

2.5 Must know how to prune insignificant cached files

2.6 Must not put undue burden on ctld for regenerating cached files

2.7 Must handle same flush request that is an update to an existing request

2.8 Must work equally across all platforms and web servers

3 Proposal: Modify cache manager to regenerate cached files

3.1 In FlushFile

3.1.1 Save stat structure for passing to TakeAction()

3.1.2 If "IsValidCurl", capture char * result to pass to TakeAction()

3.1.3 Else, pass NULL for CURL string to TakeAction

3.2 Modify TakeAction() where it handles RENAME requests

3.2.1 If the last access time (using incoming stat structure) is outside the time window specified by either config param or protocol override, keep doing what we do now (plain old rename)

3.2.2 Otherwise, IF curl string passed in is not null:

3.2.2.1 reuse ParseCurl() (from clients/common/curlUtil.c) to get the pieces of the Curl file.

3.2.2.2 Need to further parse bcaps field to grab language field

3.2.2.3 Use bcaps field to map the hash value to the set of symbols out of the metafile (if it's "00", do a no-op)

3.2.2.3.1 On a template update/create, we have the physical symbols to be used in the page generator protocol b/c the metafile entry is part of what cmd is being told to update.

3.2.2.3.2 If not a template update (meaning, someone is using [CLEAR_CACHE] from within a template to flush the cache), we have to go to the metafile in the metafiles cache to lookup the symbols.

3.2.3 ELSE it's just a file, (the path was not a CURL, just a plain old file, e.g., index.html)

3.2.4 Call the function (generatePage()) to generate a request to the page generator and save a successful request to a tempfile

3.2.5 Perform rename from temp file to old filename

3.3 GeneratePage() needs to be added

3.3.1 Functionally equivalent to clients/common/generatePageImpl.cpp

3.3.2 Input: (*this is only for the ctld page generator*)

3.3.2.1 Need the filename (CURL, including template path, or plain file) modulo the docroot path, i.e., the URI

3.3.2.2 Needs the original Curl segment, if it is a CURL (pointer that went into TakeAction())

3.3.2.3 If it exists, needs the Content id out of the CURL

3.3.3 Output:

3.3.3.1 Two filenames: from and to

3.3.3.2 If there was an error along the way return the filename and its backup equivalent

3.3.3.3 Else return the temp filename generated and the original filename

3.3.4 Create an HTTP request to the page generator (*using clients/mt_nsapi/emulate404.c as a model, refer to http://hydra.vignette.com:8080/Interfaces/DOCUMENTS/Architecture_Info/ctld-protocol.html for explanation of the settings*)

3.3.4.1 Specify a POST to '/'

3.3.4.2 Add the HTTP header stuff

3.3.4.2.1 Our content-type is application/x-www-form-urlencoded

3.3.4.2.2 Don't forget the content-length that is the size of the entity body (following the double newline that separates headers from post bodies)

3.3.4.3 Append the following in URI encoding format (name=value&name2=value2...)

3.3.4.3.1 socket=true

3.3.4.3.2 cache=auto

3.3.4.3.3 if there was a curl, curl=/0,...

3.3.4.3.4 for all bcaps symbols, the format is browser=<symbol>&browser=<symbol>&...

3.3.4.3.5 path=<templatepath>

3.3.4.3.6 content-type=text/html (unless we have CMD_MIMETYPE setting, in which case we have to request for a mapping) really what's needed is test/html or application/octet-stream the page gen only cares about it for the purposes of inserting the magic string..., so here we can fudge. By this we mean if the generated page ends in htm or html (notice the lack of . so shtml can be caught as well), then specify text/html, else use application/octet-stream in case it's a graphic or some other file that doesn't get the magic string prepended.

3.3.4.3.7 For personalization settings, we need to pass the appropriate name-val pairs used by the CURL and COMPONENT commands. (autourl-tracking, etc.)

3.3.4.3.8 Add pzn_url_exclude from config info

3.3.4.3.9 template=<template id> (*this is optional*)

3.3.4.3.10 oid=<content id>

3.3.5 Number of simultaneous page generator requests should be throttled (to avoid overburdening)

3.3.5.1 Counting semaphore on page generator requests

3.3.6 Call the page generator

3.3.7 If the response is a 200 only (i.e., if it is an OK response), save it to a temp file, and return the temp filename

3.3.8 Else, return the orig filename and backup version of it

3.3.9 The threading model in the cache manager is such that per path a thread is performing the flush, so nothing too fancy needs to happen on the temp filename in the same directory as where you're performing the rename (that also limits the clashing possibilities)

4 Other thoughts

4.1 Use ACE when appropriate

4.2 Config settings added to this:

4.2.1 Number of simultaneous regeneration requests to perform

4.2.2 Time window for what determines old versus new; think of the setting is inclusive for new, and make it seconds so comparisons are easy against st_atime

4.3 What's needed in the short term (i.e. for the customer in question):

4.3.1 Don't worry about browser caps (template variations); the customer in question isn't using them

4.3.2 Don't worry about how a time window can be specified in the flush protocol (in fact, no changes to the incoming flush request protocol should be done)

4.3.3 Keep the HTTP headers simple for now.

4.3.4 Do what makes sense as in all cases, but think in terms of this code lasting for awhile

5 Issues

5.1 *In 5.5,* Cobra issue: ***Need to switch on template type to know which page generator (and which protocol) to use to make the request.***

5.2 ***Need to optimize, file checking to leverage metafile data.***

5.3 ***RENAME and DELETE aren't granular enough for our actions:***

5.3.1 We need to distinguish between template destruction ("DESTROY") requiring cache cleanup versus don't bother with a backup file for "old" cached files ("DELETE")

5.4 ***Need to manage virtual hosting requests: at some point we need to know the domain prefix mapping, so we need the prefix piece...***

5.5 ***Need to get server-side values for HTTP headers sent to page generator representing the Server-side CGI variables***

5.6 ***CTLD configs must allow CMD to talk to it, so if you have allowable IPs and NAMEs restricted, this needs to be opened up to the CMD (or set of CMDs)***

5.7 ***In the 5.x and above releases, new references will need to be added for CMD to continue to do this...***

5.8 ***Do we care if it's for a component? Not yet...***